

**COURSE PROGRAMME****1. Information about the programme**

1.1 University	University "Alexandru Ioan Cuza" of Iasi
1.2 Faculty	Faculty of Computer Science
1.3 Department	Department of Computer Science
1.4 Domain	Computer Science
1.5 Cycle	Masters
1.6 Programme / Qualification	Studii avansate în informatică/ Advanced Studies in Computer Science

2. Information about the course

2.1 Course Name	Quality of software systems						
2.2 Course taught by	Lecturer PhD. VLAD TUDOR RADULESCU						
2.3 Seminary / laboratory taught by	Lecturer PhD. VLAD TUDOR RADULESCU						
2.4 Year	I	2.5 Semester	II	2.6 Type of evaluation *	E	2.7 Course type **	Op

*E – Exam / C – Colloquium / V – Verification

**OB – Obligatory / OP – Optionally / F – Facultative

3. Total hours (estimated per semester and activities)

3.1 Number of hours per week	4	3.2 course	2	3.3 seminary/laboratory	2
3.4 Total number of hours	56	3.5 course	28	3.6 seminary/laboratory	28
Distribution					hours
Individual study using textbooks, course notes, bibliography items, etc.					30
Supplementary study (library, on-line platforms, etc.)					30
Individual study for seminary/laboratory, homeworks, projects, etc.					45
Tutoring					5
Examination					4
Other activities					5
3.7 Total hours of individual activity *					119
3.8 Total hours per semester					175
3.9 Credit points					7

4. Pre-requisites - Curriculum (if necessary)

-

5. Conditions (if necessary)

5.1 Course	-
5.2 Seminary / Laboratory	Attending the seminary/laboratory classes is mandatory.

6. Objectives

Understanding the main elements that define the quality of software systems.
Getting acquainted with the methods used in program testing and analysis.

7. Specific competencies/Learning outcomes

- Identify the proper methodologies for developing software systems.
- Identify the proper models and methods for solving real-life problems.
- Design and implement dedicated informatic projects.

8. Contents

8.1 Course	Teaching methods	Remarks (number of hours, references)
Introduction.	exposition, debate, case studies, problem solving	-
Program testing.	exposition, debate, case studies, problem solving	-
Defects in software systems. Code inspection.	exposition, debate, case studies, problem solving	-
Risk analysis. Test planning.	exposition, debate, case studies, problem solving	-
Testing levels: unit testing, integration testing, system testing, acceptance testing.	exposition, debate, case studies, problem solving	-
Extreme testing. Regressive testing.	exposition, debate, case studies, problem solving	-
Assertions. Debugging.	exposition, debate, case studies, problem solving	-
Recapitulation.	exposition, debate, case studies, problem solving	-
Measuring the software quality. Metrics for software quality. Defect removal.	exposition, debate, case studies, problem solving	-
Measuring the software quality. Metrics for software quality. Defect removal.	exposition, debate, case studies, problem solving	-
Software reliability models.	exposition, debate, case studies, problem solving	-
Process metrics for testing.	exposition, debate, case studies, problem solving	-
Complexity metrics.	exposition, debate, case studies, problem solving	-
Recapitulation.	exposition, debate, case studies, problem solving	-

Bibliography

R. D. Craig, S. P. Jaskiel, Systematic Software Testing, SQE Publishing, 2007.
S. H. Kahn, Metrics and Models in Software Quality Engineering, Second Edition, Addison-Wesley, 2003.
Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools, Addison-Wesley, 2000.
G. J. Myers, The Art of Software Testing, Second Edition, Wiley, 2004.

8.2 Seminary / Laboratory	Teaching methods	Remarks (number of hours, references)
Program testing. Possible defects.	debate, case studies, problem solving	-
Equivalence classes.	debate, case studies, problem solving	-
Unit testing; using unit testing and mocking tools.	debate, case studies, problem solving	-
Unit testing; using unit testing and mocking tools.	debate, case studies, problem solving	-
Unit testing; using unit testing and mocking tools.	debate, case studies, problem solving	-
Load testing. Stress testing.	debate, case studies, problem solving	-
Using assertions in the testing process. Project description.	debate, case studies, problem solving	-
Recapitulation.	debate, case studies, problem solving	-
Project work - specifications.	problem solving	-
Project work - application development.	problem solving	-

8.2 Seminary / Laboratory	Teaching methods	Remarks (number of hours, references)
Project work - application development.	problem solving	-
Project work - unit testing.	problem solving	-
Project work - assertions.	problem solving	-
Project work - documentation.	problem solving	-

Bibliography

G. J. Myers, The Art of Software Testing, Second Edition, Wiley, 2004.

9. Coordination of the contents with the expectations of the community representatives, professional associations and relevant employers in the corresponding domain

10. Assessment and examination

10.1 Continuous assessment		Percentage (min. 30%)	50	
Course	Assessment type			
	Percentage		0	
	Failure to pass the continuous assessment results in failure to pass the final assessment			
	Assessment methods	Details	Percentage	with reexamination
Seminary / Laboratory	Assessment type		Practical assessment	
	Percentage		100	
	Failure to pass the continuous assessment results in failure to pass the final assessment			
	Assessment methods	Details	Percentage	with reexamination
		Continuous practical assessment	50	No
	Project	50	No	

10.2 Final assessment		Percentage (max. 70%)	50	
		Assessment type	Final written assessment	

10.3 Special notes (special situations in assessment)

-				
---	--	--	--	--

10.4 Minimum performance standard

Understanding the concepts related to software quality.
The ability to handle the development and testing of large software projects.

Date,
Course coordinator,
Lecturer PhD. VLAD TUDOR RADULESCU

Seminary coordinator,
Lecturer PhD. VLAD TUDOR RADULESCU

Approval date in the department,

Head of the department,
Assoc. Prof. PhD. ANDREI ARUSOAIIE